Computing Exact Solutions of Consensus Halving and the Borsuk-Ulam Theorem

Argyrios Deligkas
Department of Computer Science, University of Liverpool, Liverpool, UK
Leverhulme Research Centre for Functional Materials Design, Liverpool, UK
argyrios.deligkas@liv.ac.uk

John Fearnley
Department of Computer Science, University of Liverpool, Liverpool, UK
john.fearnley@liv.ac.uk

Themistoklis Melissourgos
Department of Computer Science, University of Liverpool, Liverpool, UK
t.melissourgos@liv.ac.uk

Paul G. Spirakis
Department of Computer Science, University of Liverpool, Liverpool, UK
Computer Engineering and Informatics Department, University of Patras, Patras, Greece
p.spirakis@liv.ac.uk

Abstract

We study the problem of finding an exact solution to the consensus halving problem. While recent work has shown that the approximate version of this problem is PPA-complete [29, 28], we show that the exact version is much harder. Specifically, finding a solution with n cuts is FIXP-hard, and deciding whether there exists a solution with fewer than n cuts is ETR-complete. We also give a QPTAS for the case where each agent's valuation is a polynomial.

Along the way, we define a new complexity class BU, which captures all problems that can be reduced to solving an instance of the Borsuk-Ulam problem exactly. We show that FIXP $\subseteq$ BU $\subseteq$ TFETR and that LinearBU = PPA, where LinearBU is the subclass of BU in which the Borsuk-Ulam instance is specified by a linear arithmetic circuit.

2012 ACM Subject Classification Theory of computation → Complexity classes

Keywords and phrases PPA, FIXP, ETR, consensus halving, circuit, reduction, complexity class

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.291

Related Version A full version of the paper is available at http://arxiv.org/abs/1903.03101.

Funding *John Fearnley*: The work of this author was supported by the EPSRC grant EP/P020909/1.
Paul G. Spirakis: The work of this author was supported by the EPSRC grant EP/P02002X/1.

1 Introduction

Dividing resources among agents in a fair manner is among the most fundamental problems in multi-agent systems [16]. Cake cutting [6, 8, 7, 15], and rent division [14, 33, 25] are prominent examples of problems that lie in this category. At their core, each of these problems has a desired solution whose existence is usually proved via a theorem from algebraic topology such as Brouwer's fixed point theorem, Sperner's lemma, or Kakutani's fixed point theorem.

In this paper, we focus on a fair-division problem called *consensus-halving*: an object A represented by $[0, 1]$ is to be divided into two halves A_+ and A_-, so that n agents agree that A_+ and A_- have the same value. Provided the agents have bounded and continuous valuations over A, this can always be achieved using at most n cuts, and this fact can be proved via the Borsuk-Ulam theorem from algebraic topology [44]. The necklace splitting

and ham-sandwich problems are two other examples of fair-division problems for which the existence of a solution can be proved via the Borsuk-Ulam theorem [4, 5, 37].

Recent work has further refined the complexity status of *approximate* consensus halving, in which we seek a division of the object so that every agent agrees that the values of A_+ and A_- differ by at most ϵ. Since the problem always has a solution, it lies in TFNP, which is the class of function problems in NP that always have a solution. More recent work has shown that the problem is PPA-complete [29], even for ϵ that is inverse-polynomial in n [28]. The problem of deciding whether there exists an approximate solution with k-cuts when $k < n$ is NP-complete [27]. These results are particularly notable, because they identify consensus halving as one of the first natural PPA-complete problems.

While previous work has focused on approximate solutions to the problem, in this paper we study the complexity of solving the problem *exactly*. For problems in the complexity class PPAD, which is a subclass of both TFNP and PPA, prior work has found that there is a sharp contrast between exact and approximate solutions. For example, the Brouwer fixed point theorem is the theorem from algebraic topology that underpins PPAD. Finding an approximate Brouwer fixed point is PPAD-complete [37], but finding an exact Brouwer fixed point is complete for (and the defining problem of) a complexity class called FIXP [26].

It is believed that FIXP is significantly harder than PPAD. While PPAD $\subseteq$ TFNP $\subseteq$ FNP, there is significant doubt about whether FIXP $\subseteq$ FNP. The reason for this is that there are Brouwer instances for which all solutions are irrational. This is not particularly relevant when we seek an approximate solution, but is a major difficulty when we seek an exact solution. For example, the square-root-sum problem asks us to decide for integers a_1, a_2, $\ldots$, a_n, t, whether $\sum_{i=1}^{n} \sqrt{a_i} \leq t$. This deceptively simple problem is not known to lie in NP, and can be reduced to the problem of finding an exact Brouwer fixed point [26], which provides evidence that FIXP may be significantly harder than FNP.

Our contribution. In this paper, we study the complexity of solving the consensus halving problem exactly. In our formulation of the problem, the valuation function of the agents is presented as an arbitrary arithmetic circuit, and the task is to cut A such that all agents agree that A_+ and A_- have exactly the same valuation. We study two problems. The (n, n)-Consensus Halving problem asks us to find an exact solution for n-agents using at most n-cuts, while the (n, k)-Consensus Halving problem asks us to decide whether there exists an exact solution for n-agents using at most k-cuts, where $k < n$.

Our results for (n, n)-Consensus Halving are intertwined with a new complexity class that we call BU. This class consists of all problems that can be reduced in polynomial time to the problem of finding a solution of the Borsuk-Ulam problem. We show that (n, n)-Consensus Halving lies in BU, and is FIXP hard. The hardness for FIXP implies that the exact variant of consensus halving is significantly harder than the approximate variant: while the approximate problem is PPA-complete, the exact variant is unlikely to be in FNP.

We show that (n, k)-Consensus Halving is ETR-complete. The complexity class ETR consists of all decision problems that can be formulated in the *existential theory of the reals*. It is known that NP $\subseteq$ ETR $\subseteq$ PSPACE [17], and it is generally believed that ETR is distinct from the other two classes. So our result again shows that the exact version of the problem seems to be much harder than the approximate version, which is NP-complete [27].

Just as FIXP can be thought of as the exact analogue of PPAD, we believe that BU is the exact analogue of PPA, and we provide some evidence to justify this. It has been shown that LinearFIXP = PPAD [26], which is the version of the class in which arithmetic circuits are restricted to produce piecewise *linear* functions (FIXP allows circuits to compute piecewise polynomials). We likewise define LinearBU, which consists of all problems that can be

reduced to a solution of a Borsuk-Ulam problem using a piecewise linear function, and we show that LinearBU = PPA.

The containment LinearBU ⊆ PPA can be proved using similar techniques to the proof that LinearFIXP ⊆ PPAD. However, the proof that PPA ⊆ LinearBU utilises our BU containment result for consensus halving. In particular, when the input to the consensus halving problem is a piecewise linear function, our containment result shows that the problem actually lies in LinearBU. The PPA-hardness results for consensus halving show that piecewise-linear-consensus halving is PPA-hard, which completes the containment [29, 28].

Finally, we show that, for the case where each agent has a non-piecewise polynomial valuation of constant (resp. $O(\log n)$) degree, an approximate solution to the problem can be found using $O(\log n)$ (resp. $O(poly \log n)$) cuts, which then yields a QPTAS for the problem.

For detailed proofs of the results presented, we refer the reader to the full version [22].

Related work. Although for a long period there were a few results about PPA, recently there has been a flourish of PPA-completeness results. The first PPA-completeness result was given by [32] who showed PPA-completeness of the Sperner problem for a non-orientable 3-dimensional space. In [30] this result was strengthened for a non-orientable and locally 2-dimensional space. In [3], 2-dimensional Tucker was shown to be PPA-complete; this result was used in [29, 28] to prove PPA-completeness for approximate consensus halving. In [23] PPA-completeness was proven for a special version of Tucker and for problems of the form "given a discrete fixed point in a non-orientable space, find another one". Finally, in [24] it was shown that octahedral Tucker is PPA-complete. In [35], a subclass of 2DLinearFIXP ⊆ FIXP that consists of 2-dimensional fixed-point problems was studied, and it was proven that 2DLinearFIXP = PPAD.

A large number of problems are now known to be ETR-complete: geometric intersection problems [34, 39], graph-drawing problems [1, 9, 18, 40], matrix factorization problems [42, 43], the Art Gallery problem [2], and deciding the existence of constrained (symmetric) Nash equilibria in (symmetric) normal form games with at least three players [10, 11, 12, 13, 31].

2 Preliminaries

2.1 Arithmetic circuits

An arithmetic circuit represents a continuous function $f : \mathbb{R}^n \to \mathbb{R}^m$, and is defined by a pair $(V, \mathcal{T})$, where V is a set of nodes and $\mathcal{T}$ is a set of gates. There are n nodes in V that are *input nodes*, and m nodes in V that are *output nodes*. When a value $x \in \mathbb{R}^n$ is presented at the input nodes, the circuit computes values for all other nodes $v \in V$, which we will denote as $x[v]$. The values of $x[v]$ for the m output nodes determine the value of $f(x) \in \mathbb{R}^m$.

Every node in V, other than the input nodes, is required to be the output of exactly one gate in $\mathcal{T}$. Each gate $g \in \mathcal{T}$ enforces an arithmetic constraint on its output node, based on the values of some other node in the circuit. Cycles are not allowed in these constraints. We allow the operations $\{\zeta, +, -, *\zeta, *, \max, \min\}$, which correspond to the gates shown in Table 1. Note that every gate computes a continuous function over its inputs, and thus any function f that is represented by an arithmetic circuit of this form is also continuous.

We study two types of circuits in this paper. *General* arithmetic circuits are allowed to use any of the gates that we have defined above. *Linear* arithmetic circuits allow only the operations $\{\zeta, +, -, *\zeta, \max, \min\}$, and the $*$ operation (multiplication of two variables) is disallowed. Observe that a linear arithmetic circuit computes a piecewise linear function.

Gate	Constraint
$G_\zeta(\zeta, v_{out})$	$x[v_{out}] = \zeta$, where $\zeta \in \mathbb{Q}$
$G_+(v_{in1}, v_{in2}, v_{out})$	$x[v_{out}] = x[v_{in1}] + x[v_{in2}]$
$G_-(v_{in1}, v_{in2}, v_{out})$	$x[v_{out}] = x[v_{in1}] - x[v_{in2}]$
$G_{*\zeta}(\zeta, v_{in}, v_{out})$	$x[v_{out}] = x[v_{in1}] \cdot \zeta$, where $\zeta \in \mathbb{Q}$
$G_*(v_{in1}, v_{in2}, v_{out})$	$x[v_{out}] = x[v_{in1}] \cdot x[v_{in2}]$
$G_{\max}(v_{in1}, v_{in2}, v_{out})$	$x[v_{out}] = \max\{x[v_{in1}], x[v_{in2}]\}$
$G_{\min}(v_{in1}, v_{in2}, v_{out})$	$x[v_{out}] = \min\{x[v_{in1}], x[v_{in2}]\}$

Table 1 The types of gates and their constraints.

2.2 The Consensus Halving problem

In the consensus halving problem there is an object A that is represented by the $[0, 1]$ line segment, and there are n agents. We wish to divide A into two (not necessarily contiguous) pieces such that every agent agrees that the two pieces have equal value. Simmons and Su [44] have shown that, provided the agents have bounded and continuous valuations over A, then we can find a solution to this problem using at most n cuts.

In this paper we consider instances of the consensus halving problem where the valuations of the agents are presented as arithmetic circuits. Each agent has a valuation function $f_i : [0, 1] \to \mathbb{R}$, but it is technically more convenient if they give us a representation of the *integral* of this function. So for each agent i, we are given an arithmetic circuit computing $F_i : [0, 1] \to \mathbb{R}$ where for all $x \in [0, 1]$ we have $F_i(x) = \int_0^x f(y)\, dy$. Then, the value of any particular segment of $[a, b]$ to agent i can be computed as $F_i(b) - F_i(a)$.

A solution to the consensus halving problem is given by a *k-cut* of the object A, which is defined by a vector of *cut-points* $(t_1, t_2, \ldots, t_k) \in [0, 1]^k$, and a vector of *signs* $(s_1, s_2, \ldots, s_{k+1}) \in \{-1, +1\}^{k+1}$. The cut-points t_i split A into up to $k + 1$ pieces. Note that they may in fact split A into fewer than $k + 1$ pieces in the case where two cut-points $t_i = t_j$ overlap. We define X_i to be the ith piece of A, meaning that $X_0 = [0, t_1]$, $X_i = [t_i, t_{i+1}]$ for all i in the range $1 \leq i < k$, and $X_k = [t_k, 1]$.

The sign vector determines which half of A the piece belongs to. We define $A_+ := \{X_i : s_i = +1\}$ and $A_- := \{X_i : s_i = -1\}$ to be the two halves. For each agent i, we denote the value A_+ to agent i as $F_i(A_+) := \sum_{[a,b] \in A_+} (F_i(b) - F_i(a))$, and we define $F_i(A_-)$ analogously. The k-cut is a solution to the consensus halving problem if $F_i(A_+) = F_i(A_-)$ for all agents i.

We define two computational problems. Simmons and Su [44] have proved that there always exists a solution using at most n-cuts, and our first problem is to find that solution.

(n, n)-Consensus Halving

Input: For every agent $i \in [n]$, an arithmetic circuit F_i computing the integral of agent i's valuation function.
Task: Find an n-cut for A such that $F_i(A_+) = F_i(A_-)$, for every agent $i \in [n]$.

For $k < n$ a solution to the problem may or may not exist. So we define the following decision variant of the problem.

(n,k)-Consensus Halving

Input: For every agent $i \in [n]$, an arithmetic circuit F_i computing the integral of agent i's valuation function.
Task: Decide whether there exists a k-cut for A such that $F_i(A_+) = F_i(A_-)$, for every agent $i \in [n]$.

For either of these two problems, if all of the inputs are represented by linear arithmetic circuits, then we refer to the problem as Linear Consensus Halving. We note that the known hardness results [27, 29] for consensus halving fall into this class. Specifically, those results produce valuations that are piecewise constant, and so the integral of these functions is piecewise linear, and these functions can be written down as linear arithmetic circuits [36].

3 The Class BU

The Borsuk-Ulam theorem states that every continuous function from the surface of an $(d+1)$-dimensional sphere to the d-dimensional Euclidean space maps at least one pair of antipodal points to the same point.

▶ **Theorem 1** (Borsuk-Ulam). *Let $f : S^d \to \mathbb{R}^d$ be a continuous function, where S^d is a $(d+1)$-dimensional sphere. Then, there exists an $x \in S^d$ such that $f(x) = f(-x)$.*

This theorem actually works for any domain D that is antipode-preserving homeomorphism of S^d, where by "antipode-preserving" we mean that for every $x \in D$ we have that $-x \in D$. In this paper, we choose S^d to be the sphere in $d+1$ dimensions with respect to L_1 norm: $S^d := \left\{x \mid x = (x_1, x_2, \ldots, x_{d+1}),\ \sum_{i=1}^{d+1} |x_i| = 1\right\}$.

We define the *Borsuk-Ulam* problem as follows.

Borsuk-Ulam

Input: A continuous function $f : \mathbb{R}^{d+1} \to \mathbb{R}^d$ presented as an arithmetic circuit.
Task: Find an $x \in S^d$ such that $f(x) = f(-x)$.

Note that we cannot constrain an arithmetic circuit to only take inputs from the domain S^d, so we instead put the constraint that $x \in S^d$ onto the solution.

The complexity class BU is defined as follows.

▶ **Definition 2** (BU). *The complexity class* BU *consists of all search problems that can be reduced to* Borsuk-Ulam *in polynomial time.*

3.1 LinearBU

When the input to a Borsuk-Ulam instance is a linear arithmetic circuit, then we call the problem Linear Borsuk-Ulam, and we define the class LinearBU as follows.

▶ **Definition 3** (LinearBU). *The complexity class* LinearBU *consists of all search problems that can be reduced to* Linear Borsuk-Ulam *in polynomial time.*

We will show that LinearBU = PPA. The proof that LinearBU $\subseteq$ PPA is similar to the proof that Etessami and Yannakakis used to show that LinearFIXP $\subseteq$ PPAD [26], while the fact that PPA $\subseteq$ LinearBU will follow from our results on consensus halving in Section 4.

To prove LinearBU $\subseteq$ PPA we will reduce to the *approximate* Borsuk-Ulam problem. It is well known that the Borsuk-Ulam theorem can be proved via Tucker's lemma, and Papadimitriou noted that this implies that finding an approximate solution to a Borsuk-Ulam problem lies in PPA [37]. This is indeed correct, but the proof provided in [37] is for a slightly different problem[1]. Since our results will depend on this fact, we provide our own definition and self-contained proof here. We define the approximate Borsuk-Ulam problem as follows.

ϵ-Borsuk-Ulam

Input: A continuous function $f : \mathbb{R}^{d+1} \to \mathbb{R}^d$ presented as an arithmetic circuit, along with two constants $\epsilon, \lambda \in \mathbb{R}$.
Task: Find one of the following.
1. A point $x \in S^d$ such that $\|f(x) - f(-x)\|_\infty \leq \epsilon$.
2. Two points $x, y \in S^d$ such that $\|f(x) - f(y)\|_\infty > \lambda \cdot \|x - y\|_\infty$.

The first type of solution is an approximate solution to the Borsuk-Ulam problem, while the second type of solution consists of any two points that witness that the function is not λ-Lipschitz continuous in the L_∞-norm. The second type of solution is necessary, because an arithmetic circuit is capable, through repeated squaring, of computing doubly-exponentially large numbers, and the reduction to Tucker may not be able to find an approximate solution for such circuits. We now re-prove the result of Papadimitriou in the following lemma.

▶ **Lemma 4** ([37]). *ϵ-Borsuk-Ulam is in* PPA.

To show that LinearBU $\subseteq$ PPA we will provide a polynomial time reduction from Linear Borsuk-Ulam to ϵ-Borsuk-Ulam. To do this, we follow closely the technique used by Etessami and Yannakakis to show that LinearFIXP $\subseteq$ PPAD [26]. The idea is to make a single call to ϵ-Borsuk-Ulam to find an approximate solution to the problem for a suitably small ϵ, and to then round to an exact solution by solving a linear program. To build the LP, we depend on the fact that we have access to the linear arithmetic circuit that represents f.

▶ **Lemma 5.** *Linear Borsuk-Ulam is in* PPA.

4 Containment Results for Consensus Halving

4.1 (n, n)-Consensus Halving is in BU and LinearBU = PPA

We show that (n, n)-Consensus Halving is contained in BU. Simmons and Su [44] show the existence of an n-cut solution to the consensus halving problem by applying the Borsuk-Ulam theorem, and we follow their approach in this reduction. However, we must show that the approach can be implemented using arithmetic circuits. We take care in the reduction to avoid G_* gates, and so if the inputs to the problem are all linear arithmetic circuits, then our reduction will produce a Linear Borsuk-Ulam instance. Hence, we also show that (n, n)-Linear Consensus Halving is in LinearBU.

▶ **Theorem 6.** *The following two containments hold.*

[1] The problem used in [37] presents the function as a polynomial-time Turing machine rather than an arithmetic circuit, and the Lipschitzness of the function is guaranteed by constraining the values that it can take.

- (n, n)-Consensus Halving *is in* BU.
- (n, n)-Linear Consensus Halving *is in* LinearBU.

We note that this also implies that PPA $\subseteq$ LinearBU, thereby completing the proof that PPA = LinearBU. Specifically, Filos-Ratsikas and Goldberg have shown that *approximate*-(n, n)-Consensus Halving is PPA-complete, and their valuation functions are piecewise constant. Therefore, the integrals of these functions are piecewise linear, and so their approximate-(n, n)-Consensus Halving instances can be reduced to (n, n)-Linear Consensus Halving. Hence (n, n)-Linear Consensus Halving is PPA-hard, which along with Lemma 5 implies the following corollary.

▶ **Corollary 7.** PPA = LinearBU.

4.2 (n, k)-Consensus Halving is in ETR

The existential theory of the reals consists of all true existentially quantified formulae using the connectives $\{\wedge, \vee, \neg\}$ over polynomials compared with the operators $\{<, \leq, =, \geq, >\}$. The complexity class ETR captures all problems that can be reduced in polynomial time to the existential theory of the reals.

We prove that (n, k)-Consensus Halving is in ETR. The reduction simply encodes the arithmetic circuits using ETR formulas, and then constrains $F_i(A_+) = F_i(A_-)$ for every agent i.

▶ **Theorem 8.** (n, k)-Consensus Halving *is in* ETR.

Using the same technique, we can also reduce Borsuk-Ulam to an ETR formula. In this case, we get an ETR formula that always has a solution, and so this result places the problem in TFETR, which is the subclass of ETR in which the formula is guaranteed to be true.

▶ **Theorem 9.** BU $\subseteq$ TFETR.

5 Hardness Results for Consensus Halving

In this section we prove that (n, n)-Consensus Halving is FIXP-hard and that $(n, n-1)$-Consensus Halving is ETR-hard. These two reductions share a common step of embedding an arithmetic circuit into a consensus halving instance. So we first describe this step, and then move on to proving the two individual hardness results.

5.1 Embedding a circuit in a Consensus Halving instance

Our approach is inspired by [27], who provided a reduction from ϵ-GCircuit [19, 38] to approximate consensus halving. However, our construction deviates significantly from theirs due to several reasons.

Firstly, the reduction in [27] works *only* for approximate consensus halving. Specifically, some valuations used in that construction have the form of $1/\epsilon$, where ϵ is the approximation guarantee, so the construction is not well-defined when $\epsilon = 0$ as it is in our case. Many of the gate gadgets used in [27] cannot be used due to this issue, including the max gate, which is crucially used in that construction to ensure that intermediate values do not get too large. We provide our own implementations of the broken gates. Our gate gadgets only work when the inputs and outputs lie in the range $[0, 1]$, and so we must carefully construct circuits for which this is always the case. The second major difference is that the reduction in [27]

Special Gate	Constraint	Ranges
$G_{()^2}(v_{in}, v_{out})$	$x[v_{out}] = (x[v_{in}])^2$	$x[v_{in}] \in [0,1]$
$G_{*2}^{[0,1]}(v_{in}, v_{out})$	$x[v_{out}] = x[v_{in}] \cdot 2$	$x[v_{in}] \in [0,1/2]$
$G_{-}^{[0,1]}(v_{in1}, v_{in2}, v_{out})$	$x[v_{out}] = \max\{x[v_{in1}] - x[v_{in2}], 0\}$	$x[v_{in1}], x[v_{in2}] \in [0,1]$

Table 2 The special types of gates, their constraints and ranges of input.

does not provide any method of multiplying two variables, which is needed in our case. We construct a gadget to do this, based on a more primitive gadget for squaring a single variable.

Special circuit. Our reduction from an arithmetic circuit to consensus halving will use a very particular subset of gates. Specifically, we will not use $G_{\min}$, $G_{\max}$, or G_*, and we will restrict $G_{*\zeta}$ so that ζ must lie in $(0,1]$. We do however introduce three new gates, shown in Table 2. The gate $G_{()^2}$ squares its input, the gate $G_{*2}^{[0,1]}$ multiplies its input by two, but requires that the input be in $[0,1/2]$, and the gate $G_{-}^{[0,1]}$ is a special minus gate that takes as inputs $a, b \in [0,1]$ and outputs $\max\{a-b,0\}$. We note that $G_{\min}$, $G_{\max}$, and G_* can be implemented in terms of our new gates according to the following identities.

$$\max\{a,b\} = \frac{a+b}{2} + \frac{|a-b|}{2} = \frac{a}{2} + \frac{b}{2} + \frac{1}{2}\max\{a-b,0\} + \frac{1}{2}\max\{b-a,0\},$$

$$\min\{a,b\} = \frac{a+b}{2} - \frac{|a-b|}{2} = \frac{a}{2} + \frac{b}{2} - \frac{1}{2}\max\{a-b,0\} - \frac{1}{2}\max\{b-a,0\},$$

$$a \cdot b = 2\left[\left(\frac{a}{2} + \frac{b}{2}\right)^2 - \left(\left(\frac{a}{2}\right)^2 + \left(\frac{b}{2}\right)^2\right)\right].$$

Also, a very important requirement of the special circuit is that both inputs of any G_+ gate are in $[0,1/2]$. To make sure of that, we downscale the inputs before reaching the gate, and upscale the output, using the fact that $a + b = (a/2 + b/2) \cdot 2$.

The reduction to consensus halving. The reduction follows the general outline of the reduction given in [27]. The construction is quite involved, and so we focus on the high-level picture here. Each gate is implemented by 4 agents, namely ad, mid, cen, ex in the consensus halving instance. The values computed by the gates are encoded by the positions of the cuts that are required in order to satisfy these agents. Agent ad performs the exact mathematical operation of the gate, and feeds the outcome in mid, who "trims" it in accordance with the gate's actual operation. Then mid feeds her outcome to cen and ex, who make a copy of mid's correct value of the gate, with "negative" and "positive" labels respectively. This value with the appropriate label will be input to other gates.

The most important agents are the ones that perform the mathematical operation of each gate, i.e. agents ad. Figure 1 shows the part of the valuation functions of these agents that perform the operation. Each figure shows a valuation function for one of the agents, meaning that the blue regions represent portions of the object that the agent desires. The agent's valuation for any particular interval is the integral of this function over that interval.

To understand the high-level picture of the construction, let us look at the construction for $G_{*\zeta}$. The precise valuation functions of the agents in the construction ensure that there is exactly one *input* cut in the region v_{in}^+. The leftmost piece due to that cut in that region will belong to A_+, while the rightmost will belong to A_-. It is also ensured that there is exactly one *output* cut in the region v_{out}^a, and that the first piece in that region will belong to A_- and the second will belong to A_+.

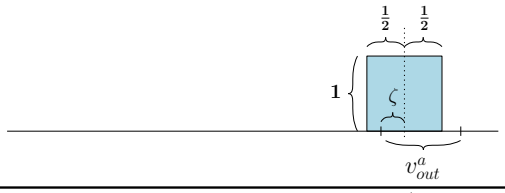
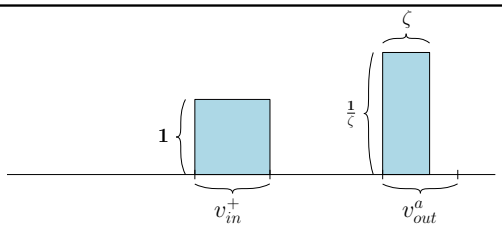
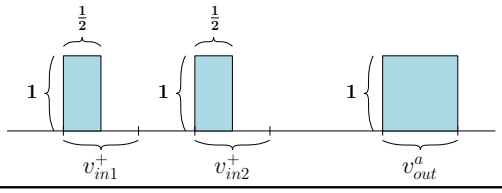
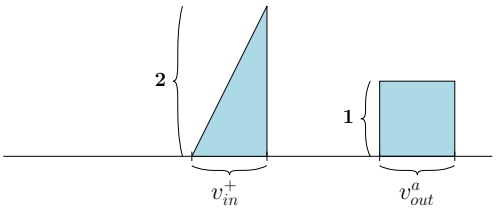
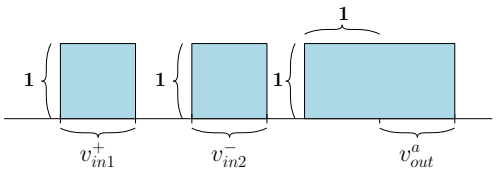
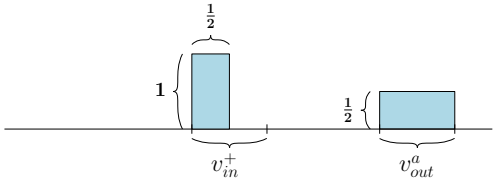
Gate	$\mathbf{G}_\pi(\mathbf{t})$	Valuation function
$\mathbf{G}_\zeta$	$\begin{cases} 1 & \textbf{if } t \in [v^a_{out,l} + \zeta - \frac{1}{2}, v^a_{out,l} + \zeta + \frac{1}{2}] \\ 0 & \textbf{otherwise} \end{cases}$	
$\mathbf{G}_{*\zeta}$	$\begin{cases} 1 & \textbf{if } t \in v^+_{in} \\ 1/\zeta & \textbf{if } t \in [v^a_{out,l}, v^a_{out,l} + \zeta] \\ 0 & \textbf{otherwise} \end{cases}$	
$\mathbf{G}_+$	$\begin{cases} 1 & \textbf{if } t \in [v^+_{in1,l}, v^+_{in1,l} + \frac{1}{2}] \\ 1 & \textbf{if } t \in [v^+_{in2,l}, v^+_{in2,l} + \frac{1}{2}] \\ 1 & \textbf{if } t \in v^a_{out} \\ 0 & \textbf{otherwise} \end{cases}$	
$\mathbf{G}_{()^2}$	$\begin{cases} 2(t - v^+_{in,l}) & \textbf{if } t \in v^+_{in} \\ 1 & \textbf{if } t \in v^a_{out} \\ 0 & \textbf{otherwise} \end{cases}$	
$\mathbf{G}^{[0,1]}_-$	$\begin{cases} 1 & \textbf{if } t \in v^+_{in1} \\ 1 & \textbf{if } t \in v^-_{in2} \\ 1 & \textbf{if } t \in [v^a_{out,l} - 1, v^a_{out,r}] \\ 0 & \textbf{otherwise} \end{cases}$	
$\mathbf{G}^{[0,1]}_{*2}$	$\begin{cases} 1 & \textbf{if } t \in [v^+_{in,l}, v^+_{in,l} + \frac{1}{2}] \\ 1/2 & \textbf{if } t \in v^a_{out} \\ 0 & \textbf{otherwise} \end{cases}$	

Figure 1 Gates and their corresponding functions $G_\pi(t)$.

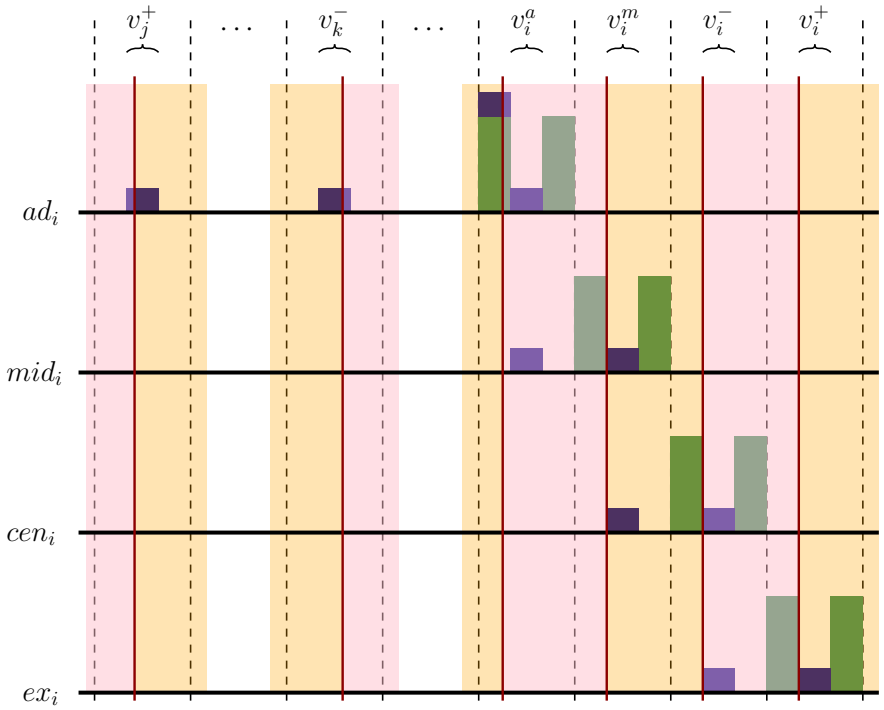


Figure 2 An example where the computation at the output $v_{out} := v_i$ of a $G_{-}^{[0,1]}$ gate with inputs $v_{in1} := v_j$ and $v_{in2} := v_k$ is simulated by the Consensus Halving instance. Here $x[v_j] = 1/4$ and $x[v_k] = 3/4$, hence $x[v_i] = 0$. The information about the values of the inputs is encoded by the cuts (red lines) in intervals v_j^+, and v_k^- imposed by agents ex_j and cen_k respectively. The blue and green shapes depict the area below the valuation function of each of the 4 agents. The pink regions have label "+" while the yellow have label "−". Agent ad_i performs the subtraction, by demanding that she is satisfied, and places a cut 1/10 to the left of the left endpoint of interval v_i^a. Then agent mid_i gets satisfied by placing a cut at exactly the left endpoint of interval v_i^m, thus encoding the value 0 which is the correct output value of the gate. Finally, agents cen_i, ex_i copy this value by enforcing similar cuts at the left endpoints of intervals v_i^- and v_i^+ respectively. The encoded values in the latter two intervals are the "negative" and "positive" version of $x[v_i]$.

Suppose that gate g_i in the circuit is of type $G_{*\zeta}$ and we want to implement it through a Consensus Halving instance. If we treat v_{in}^+ and v_{out}^a in Figure 1 as representing $[0, 1]$, then agent ad_i will take as input a cut at point $x \in v_{in}^+$. In order to be satisfied, ad_i will impose a cut at point $y \in v_{out}^a$, such that $F_i(A_+) = F_i(A_-)$, where: $F_i(A_+) = x + (\zeta - y)/\zeta$ and $F_i(A_-) = (1 - x) + y/\zeta$. Simple algebraic manipulation can be used to show that ad_i is satisfied only when $y = \zeta \cdot x$, as required.

We show that the same property holds for each of the gates in Figure 1. Two notable constructions are for the gates $G_{()^2}$ and $G_{-}^{[0,1]}$. For the gate $G_{()^2}$ the valuation function of agent ad is non-constant, which is needed to implement the non-linear squaring function. For the gate $G_{-}^{[0,1]}$, note that the output region v_{out}^a only covers half of the possible output space. The idea is that if the result of $x[v_{in1}] - x[v_{in2}]$ is negative, then the output cut will lie before the output region, which will be interpreted as a zero output by agents mid, cen, ex in the construction. On the other hand, if the result is positive, the result will lie in the usual output range, and will be interpreted as a positive number. An example where $x[v_{in1}] = 1/4$ and $x[v_{in2}] = 3/4$ is shown in Figure 2.

Ultimately, this allows us to construct a consensus-halving instance that implements this circuit. This means that for any $x \in [0,1]^n$, we can encode x as a set of cuts, which then force cuts to be made at each gate gadget that encode the correct output for that gate. The full details of the construction are quite involved, but we are able to show the following result.

▶ **Lemma 10.** *Suppose that we are given an arithmetic circuit with the following properties.*
- *The circuit uses the gates* $G_\zeta, G_+, G_{*\zeta}, G_{()^2}, G_-^{[0,1]}, G_{*2}^{[0,1]}$.
- *Every* G_ζ *and* $G_{*\zeta}$ *has* $\zeta \in \mathbb{Q} \cap (0,1]$.
- *For every input* $x \in [0,1]^n$*, all intermediate values computed by the circuit lie in* $[0,1]$.

We can construct a consensus-halving instance that implements this circuit.

5.2 (n,n)-Consensus Halving is FIXP-hard

We show that (n,n)-Consensus Halving is FIXP-hard by reducing from the problem of finding a Nash equilibrium in a d-player game, which is known to be a FIXP-complete [26]. As shown in [26], this problem can be reduced to the Brouwer fixed point problem: given an arithmetic circuit computing a function $F : [0,1]^n \to [0,1]^n$, find a point $x \in [0,1]^n$ such that $F(x) = x$. In a similar way to [27], we take this circuit and embed it into a consensus halving instance, with the outputs looped back to the inputs. Since Lemma 10 implies that our implementation of the circuit is correct, this means that any solution to the consensus halving problem must encode a point x satisfying $F(x) = x$.

One difficulty is that we must ensure that the arithmetic circuit that we build falls into the class permitted by Lemma 10. To do this, we carefully analyse the circuits produced in [26], and we modify them so that all of the preconditions of Lemma 10 hold.

▶ **Theorem 11.** (n,n)-Consensus Halving *is* FIXP*-hard.*

This theorem, along with Theorem 6 give the following corollary.

▶ **Corollary 12.** FIXP $\subseteq$ BU.

5.3 $(n,n-1)$-Consensus Halving is ETR-complete

We will show the ETR-hardness of $(n,n-1)$-Consensus Halving by reducing from the following problem Feasible, which is known to be ETR-complete [41].

▶ **Definition 13** (Feasible, Feasible$_{[0,1]}$)**.** *Let* $p(x_1,\ldots,x_m)$ *be a polynomial.* Feasible *asks whether there exists a point* $(x_1,\ldots,x_m) \in \mathbb{R}^m$ *that satisfies* $p(x_1,\ldots,x_m) = 0$. Feasible$_{[0,1]}$ *asks whether there exists a point* $(x_1,\ldots,x_m) \in [0,1]^m$ *that satisfies* p.

The idea is to turn the polynomial into a circuit, and then embed that circuit into a consensus halving instance using Lemma 10. As before, the main difficulty is ensuring that the preconditions of Lemma 10 are satisfied. To do this, we must ensure that the the inputs to the circuit take values in $[0,1]$, which is not the case if we reduce directly from Feasible. Instead, we first consider the problem Feasible$_{[0,1]}$, in which x is constrained to lie in $[0,1]^n$ rather than $\mathbb{R}^n$, and we show the following result.

▶ **Lemma 14.** Feasible$_{[0,1]}$ *is* ETR*-complete.*

ETR$_{[0,1]}$ is the subclass of ETR in which variables are quantified over $[0,1]^n$ rather than $\mathbb{R}^n$. The above lemma follows from the fact that ETR$_{[0,1]}$ = ETR, and the fact that Feasible$_{[0,1]}$

is $\texttt{ETR}_{[0,1]}$-hard. This equivalence of classes, together with the completeness of $\textsc{Feasible}_{[0,1]}$ may be of independent interest.

We then proceed to reduce $\textsc{Feasible}_{[0,1]}$ to $(n, n-1)$-Consensus Halving. We still don't quite meet the requirements of Lemma 10, because the intermediate terms may be outside $[0, 1]$. We resolve this by implementing a circuit $p^+(x)$ implementing only the positive terms of $p(x)$ downscaled appropriately, and a circuit $p^-(x)$ implementing the positive terms of $-p(x)$ again downscaled appropriately. The check agent is then satisfied if $p^+(x) = p^-(x)$, which can only occur when $p(x) = 0$.

There will be $n-1$ *choice* agents corresponding to the $(n-1)/4$ nodes of the circuit, who enforce that there is a cut for each of the nodes to the circuit, and together these cuts encode an input x to the polynomial. Each agent introduced by Lemma 10 has an associated cut that is forced by the construction used in that lemma, and these cuts compute the output of the associated gate.

So far, every agent has a corresponding cut that is forced by the construction. There is, however, one final *check* agent who has the following properties.

- If $p(x) = 0$, then the check agent agrees that A has been cut in half without an extra cut being made.
- If $p(x) \neq 0$, then the check agent requires one more cut to be made in order to be satisfied that A has been cut in half.

Hence, if there is a solution to Feasible, then there is a solution to $\textsc{Feasible}_{[0,1]}$, and there is a $(n-1)$-cut that solves the Consensus Halving instance. Otherwise there is no such solution.

▶ **Theorem 15.** $(n, n-1)$-Consensus Halving *is* ETR*-complete.*

6 A QPTAS for Consensus Halving with polynomial valuation functions

In this section we show that an approximate solution to the consensus halving problem can be found in quasi-polynomial time when each agent's valuation function is a single polynomial of constant or even polylogarithmic degree. We will do so by formulating the problem as a formula in the *approximate* existential theory of the reals, and then applying the approximation theorem proved in [20, 21].

Our result implies that these instances can be solved approximately using a polylogarithmic number of cuts. We note that this is one of the most general classes of instances for which we could hope to prove such a result: any instance in which n agents desire completely disjoint portions of the object can only be solved by an n-cut, and piecewise linear functions are capable of producing such a situation. So in a sense, we are exploiting the fact that this situation cannot arise when the agents have non-piecewise polynomial valuation functions.

▶ **Lemma 16.** *For every* Consensus Halving *instance with n agents, and every $\epsilon > 0$, if each agent's valuation function F_i is a single polynomial of degree at most $O(poly \log n)$, then there exists a k-cut, where $k := O(poly \log n)/\epsilon^4$, and pieces A_+ and A_- such that:*

- *every cut point is a multiple of $1/k = \frac{\epsilon^4}{O(poly \log n)}$;*
- *$|F_i(A_+) - F_i(A_-)| \leq \epsilon$, for every agent i.*

As a consequence, we can perform a brute force search over all possible k-cuts to find an approximate solution, which can be carried out in $n^{O(poly \log n/\epsilon^4)}$ time.

▶ **Theorem 17.** Consensus Halving *admits a QPTAS when the valuation function of every agent is a single polynomial of degree $O(poly \log n)$.*

References

1 Zachary Abel, Erik D Demaine, Martin L Demaine, Sarah Eisenstat, Jayson Lynch, and Tao B Schardl. Who needs crossings? Hardness of plane graph rigidity. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 51. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

2 Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. The art gallery problem is ETR-complete. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 65–73. ACM, 2018.

3 James Aisenberg, Maria Luisa Bonet, and Sam Buss. 2-d Tucker is PPA complete. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:163, 2015.

4 Noga Alon. Splitting necklaces. *Advances in Mathematics*, 63(3):247–253, 1987.

5 Noga Alon and Douglas B West. The Borsuk-Ulam theorem and bisection of necklaces. *Proceedings of the American Mathematical Society*, 98(4):623–628, 1986.

6 Georgios Amanatidis, George Christodoulou, John Fearnley, Evangelos Markakis, Christos-Alexandros Psomas, and Eftychia Vakaliou. An improved envy-free cake cutting protocol for four agents. In *International Symposium on Algorithmic Game Theory*, pages 87–99. Springer, 2018.

7 Haris Aziz and Simon Mackenzie. A discrete and bounded envy-free cake cutting protocol for any number of agents. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 416–427. IEEE, 2016.

8 Haris Aziz and Simon Mackenzie. A discrete and bounded envy-free cake cutting protocol for four agents. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 454–464. ACM, 2016.

9 Daniel Bienstock. Some provably hard crossing number problems. *Discrete & Computational Geometry*, 6(3):443–459, 1991.

10 Vittorio Bilò and Marios Mavronicolas. The complexity of decision problems about Nash equilibria in win-lose games. In *Proc. of SAGT*, pages 37–48, 2012.

11 Vittorio Bilò and Marios Mavronicolas. Complexity of rational and irrational Nash equilibria. *Theory of Computing Systems*, 54(3):491–527, 2014.

12 Vittorio Bilò and Marios Mavronicolas. A catalog of EXISTS-R-complete decision problems about Nash equilibria in multi-player games. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 47, 2016.

13 Vittorio Bilò and Marios Mavronicolas. Existential-R-complete decision problems about symmetric Nash equilibria in symmetric multi-player games. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 66, 2017.

14 Steven J Brams and D Marc Kilgour. Competitive fair division. *Journal of Political Economy*, 109(2):418–443, 2001.

15 Steven J Brams and Alan D Taylor. An envy-free cake division protocol. *The American Mathematical Monthly*, 102(1):9–18, 1995.

16 Steven J Brams and Alan D Taylor. *Fair Division: From cake-cutting to dispute resolution*. Cambridge University Press, 1996.

17 John Canny. Some algebraic and geometric computations in PSPACE. In *Proc. of STOC*, pages 460–467, New York, NY, USA, 1988. ACM. URL: http://doi.acm.org/10.1145/62212.62257, doi:10.1145/62212.62257.

18 Jean Cardinal and Udo Hoffmann. Recognition and complexity of point visibility graphs. *Discrete & Computational Geometry*, 57(1):164–178, 2017.

19 Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player Nash equilibria. *Journal of the ACM (JACM)*, 56(3):14, 2009.

20 Argyrios Deligkas, John Fearnley, Themistoklis Melissourgos, and Paul G. Spirakis. Approximating the existential theory of the reals. In George Christodoulou and Tobias Harks, editors, *Web and Internet Economics*, pages 126–139, Cham, 2018. Springer International Publishing.

21 Argyrios Deligkas, John Fearnley, Themistoklis Melissourgos, and Paul G. Spirakis. Approximating the existential theory of the reals. *CoRR*, abs/1810.01393, 2018. URL: http://arxiv.org/abs/1810.01393, arXiv:1810.01393.

22 Argyrios Deligkas, John Fearnley, Themistoklis Melissourgos, and Paul G. Spirakis. Computing exact solutions of consensus halving and the Borsuk-Ulam theorem. *CoRR*, abs/1903.03101, 2019. URL: http://arxiv.org/abs/1903.03101, arXiv:1903.03101.

23 Xiaotie Deng, Jack R Edmonds, Zhe Feng, Zhengyang Liu, Qi Qi, and Zeying Xu. Understanding PPA-completeness. In *Proceedings of the 31st Conference on Computational Complexity*, page 23. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.

24 Xiaotie Deng, Zhe Feng, and Rucha Kulkarni. Octahedral Tucker is PPA-complete. In *Electronic Colloquium on Computational Complexity Report TR17-118*, 2017.

25 Francis Edward Su. Rental harmony: Sperner's lemma in fair division. *The American mathematical monthly*, 106(10):930–942, 1999.

26 K. Etessami and M. Yannakakis. On the complexity of Nash equilibria and other fixed points. *SIAM Journal on Computing*, 39(6):2531–2597, 2010. URL: https://doi.org/10.1137/080720826, arXiv:https://doi.org/10.1137/080720826, doi:10.1137/080720826.

27 Aris Filos-Ratsikas, Søren Kristoffer Stiil Frederiksen, Paul W. Goldberg, and Jie Zhang. Hardness results for consensus-halving. In *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, pages 24:1–24:16, 2018.

28 Aris Filos-Ratsikas and Paul W Goldberg. The complexity of splitting necklaces and bisecting ham sandwiches. *arXiv preprint arXiv:1805.12559*, 2018.

29 Aris Filos-Ratsikas and Paul W. Goldberg. Consensus halving is PPA-complete. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 51–64, 2018.

30 Katalin Friedl, Gábor Ivanyos, Miklos Santha, and Yves F Verhoeven. Locally 2-dimensional Sperner problems complete for the polynomial parity argument classes. In *Italian Conference on Algorithms and Complexity*, pages 380–391. Springer, 2006.

31 Jugal Garg, Ruta Mehta, Vijay V Vazirani, and Sadra Yazdanbod. ETR-completeness for decision versions of multi-player (symmetric) Nash equilibria. *ACM Transactions on Economics and Computation (TEAC)*, 6(1):1, 2018.

32 Michelangelo Grigni. A Sperner lemma complete for PPA. *Information Processing Letters*, 77(5-6):255–259, 2001.

33 Claus-Jochen Haake, Matthias G Raith, and Francis Edward Su. Bidding for envy-freeness: A procedural approach to n-player fair-division problems. *Social Choice and Welfare*, 19(4):723–749, 2002.

34 Jiri Matousek. Intersection graphs of segments and EXISTS-R. *arXiv preprint arXiv:1406.2636*, 2014.

35 Ruta Mehta. Constant rank bimatrix games are PPAD-hard. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 545–554, 2014. URL: https://doi.org/10.1145/2591796.2591835, doi:10.1145/2591796.2591835.

36 Sergei Ovchinnikov. Max-min representation of piecewise linear functions. *Beiträge zur Algebra und Geometrie*, 43(1):297–302, 2002. URL: http://eudml.org/doc/225460.

37 Christos H Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994.

38 Aviad Rubinstein. Inapproximability of Nash equilibrium. *SIAM Journal on Computing*, 47(3):917–959, 2018.

39 Marcus Schaefer. Complexity of some geometric and topological problems. In *International Symposium on Graph Drawing*, pages 334–344. Springer, 2009.

40 Marcus Schaefer. Realizability of graphs and linkages. In *Thirty Essays on Geometric Graph Theory*, pages 461–482. Springer, 2013.

41 Marcus Schaefer and Daniel Stefankovic. Fixed points, Nash equilibria, and the existential theory of the reals. *Theory Comput. Syst.*, 60(2):172–193, 2017. URL: https://doi.org/10.1007/s00224-015-9662-0, doi:10.1007/s00224-015-9662-0.

42 Yaroslav Shitov. A universality theorem for nonnegative matrix factorizations. *arXiv preprint arXiv:1606.09068*, 2016.

43 Yaroslav Shitov. The complexity of positive semidefinite matrix factorization. *SIAM Journal on Optimization*, 27(3):1898–1909, 2017.

44 Forest W. Simmons and Francis Edward Su. Consensus-halving via theorems of Borsuk-Ulam and Tucker. *Mathematical Social Sciences*, 45(1):15–25, 2003.